UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/468,469	12/21/1999	REGINALD V. BLUE		3674

7590 11/19/2002

ROCCO L ADORNATO
UNISYS CORP
TOWNSHIP LINE & UNION MEETING ROADS
BLUE BELL, PA 19424

EXAMINER
ALI, SYED J

ART UNIT	PAPER NUMBER
2127	

DATE MAILED: 11/19/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/468,469	Applicant(s) BLUE, REGINALD V.	
	Examiner Syed J Ali	Art Unit 2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-45* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-45* is/are rejected.

7)☒ Claim(s) *21,22,24,27 and 44* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 2 .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

Specification

1. The disclosure is objected to because of the following informalities: Reference made to Fig. 13, which does not exist (pg. 32).

Appropriate correction is required.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claim 21 recites the limitation "said access" in line 24. There is insufficient antecedent basis for this limitation in the claim.

4. Claim 21 recites the limitation "said executing act" in line 27. There is insufficient antecedent basis for this limitation in the claim.

5. Claim 22 recites the limitation "said access" in line 29. There is insufficient antecedent basis for this limitation in the claim.

6. Claim 22 recites the limitation "said executing act" in line 1. There is insufficient antecedent basis for this limitation in the claim.

7. Claim 24 recites the limitation "said executing act" in line 9. There is insufficient antecedent basis for this limitation in the claim.

8. Claim 24 recites the limitation "said claiming acts" in line 10. There is insufficient antecedent basis for this limitation in the claim.

9. Claim 27 recites the limitation "said executing act" in line 22. There is insufficient antecedent basis for this limitation in the claim.

10. Claim 44 recites the limitation "said determinations" in line 18. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

11. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

12. Claims 1, 10, and 11 are rejected under 35 U.S.C. 102(b) as being anticipated by Bertoni (USPN 5,761,659) (hereinafter Bertoni).

As per claim 1, Bertoni discloses a method of managing a resource shared among concurrently executing threads in a multi-threaded computer program running under an operating system that supports multi-threaded computer programs, said method comprising the acts of:

receiving, from a first thread, a request for a lock, said request indicating whether said request is for a read lock or a write lock (col. 5 lines 18-30, "the attribute Mode 154 has the value READ 156 which indicates that the region corresponding to this sub-lock has a read lock placed upon it...Mode could also be WRITE");

if said request is for a read lock, granting said request and permitting said thread to proceed unless another of said threads is writing said resource (col. 4 lines 50-54, "if a user has a write lock applied to a portion of the resource this means that no other user may apply a read or write lock to the same portion of the resource"); and

if said request is for a write lock, granting said request and permitting said thread to proceed unless another of said threads is reading or writing said resource (col. 4 lines 50-54, "if a user has a write lock applied to a portion of the resource this means that no other user may apply a read or write lock to the same portion of the resource. In addition, a portion of the resource that is read locked may not be write locked by another user").

As per claim 10, Bertoni discloses the method of claim 1, wherein said resource comprises a data object located within the address space of said computer program (col. 4 lines 61-67, "The resource 20 could be any resource within a computer system. By way of example, the resource 20 could be a file, a portion of a file, an object...or any other resource or media used by a computer system or network that is accessible by a user").

As per claim 11, a computer-readable medium having computer-executable instructions to perform the method of claim 1 is inherent in the disclosure of Bertoni as the method described is intended for implementation on a computer system. Thus, the system must have a medium with computer-executable instructions.

Claim Rejections - 35 USC § 103

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14. Claims 2-9, and 12-45 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bertoni in view of Gallop (USPN 6,411,983) (hereinafter Gallop).

As per claim 2, Gallop discloses a mechanism, wherein request is issued by creating a local class instance, wherein a constructor for said class instance issues said request (col. 4 lines 31-49, "when the object 2 is initially created the monitor transition

element field 3 in the object 2 stores the address or pointer 12 to the first monitor transition element 14a...each monitor transition element 14 is implemented as a structure containing two function pointers: &lock 16 and &unlock 18"). Further, since the mechanism is designed to be implemented in Java, when a class instance is created, a constructor for that class is called, and this is well known in the art. It would be obvious to one of ordinary skill in the art to combine the teachings of Gallop with those of Bertoni since the combination of these references provides a sound means of locking a resource. By using the framework of Gallop, a data structure is defined that allows encapsulation of threads and incorporates the locking and unlocking functions into that data structure. To limit this framework, the method of Bertoni ensures that no thread is reading or writing to the resource while another thread is writing to the resource. In doing so, data can be assured to be accurate, whereas if a thread was reading from the resource at the same time another was writing, the data may be compromised and could not be considered accurate.

As per claim 4, Gallop does not specifically disclose the act of destroying said local class instance, wherein a destructor for said class instance issues a request to release said lock. However, Gallop does teach that the release of a lock be performed through the use of an unlock function (col. 4 lines 59-62, "the unlock transition function 18 is invoked in response to a request to unlock an object"). It would be obvious to reference this function within the class's destructor since at the point when an object goes out of scope, it would be senseless to have a thread continue to possess an object's lock when it

no longer exists. Thus, in addition to referencing the "&unlock" function in Gallop whenever a thread desires to release a lock, it would be obvious to one of ordinary skill in the art to also reference the "&unlock" function when the object goes out of scope and the destructor is called.

As per claim 5, Gallop teaches determining whether other threads are reading or writing the resource, wherein the determination of whether other threads are reading or writing from said resource are made by claiming one or more critical sections (col. 1 lines 20-49, "because each thread will reserve resources, programming languages such as Java provide mechanisms for serializing critical sections of the program code"). It would be obvious to one of ordinary skill in the art to combine the references of Gallop and Bertoni since it would provide an advantageous method of ensuring that no thread is accessing the resource while another is writing to it. Bertoni discloses the use of a data structure to describe properties of a lock (fig. 4), but by claiming critical sections of the program code as described in Gallop, only a simple check of whether any thread possessed the write critical section would need to be performed in order to verify that no other thread is writing to the resource

As per claim 9, Gallop teaches the act of claiming at least one of said critical sections is conditioned upon the value of said counter (col. 2 lines 33-39, "the monitor transition vector...provides implicit lock counts", "this improves the execution

performance by eliminating the need to perform tests to determine if the synchronization comprises a first lock, a shallow lock or a deeply nested lock").

As per claim 12, Gallop teaches of a system for managing the use of a resource shared among concurrently-executing threads, said system comprising:

a record for maintaining information as to whether any of said threads is accessing a resource at a given point in time (col. 9 line 21 – col. 10 line 2, "the object reference field 44 holds an object reference &OBJECT-i for identifying the deeply locked object");

Gallop does not specifically disclose an object, which comprises or references:

a constructor, said constructor comprising computer-executable instructions to obtain a lock on said resource and to records said lock in said record; and

a destructor, said destructor comprising a set of computer-executable instructions to release said lock and to record the release of said lock in said record;

wherein the constructor instructions are executed upon creation of an instance of said object within a local scope, wherein the destructor instructions are executed upon the exiting of said local scope, and wherein no instruction, other than the instruction to exit said local scope, is required to release said lock.

However, Gallop does disclose an object-oriented mechanism, to be implemented in Java to perform locking. Java is well known to have class data structures defined,

which comprise a constructor, destructor, as well as other methods as needed. Gallop further teaches of computer-executable instructions comprising lock and unlock functions associated with each thread. It would have been obvious to one of ordinary skill in the art to include the code for the lock and unlock functions within the constructor and destructor, respectively, since that would allow for cleaner code management, and no programming concerns regarding allocation or freeing of resources

As per claim 15, Gallop does not specifically teach the further limitation, wherein said record comprises a counter, wherein said constructor further comprises an instruction to increment said counter, wherein said destructor comprises an instruction to decrement said counter, wherein said constructor further comprises instruction to condition the claiming of said 'critical section upon the value of said counter, and wherein said destructor further comprises instructions to condition the relinquishment of said critical section upon the value of said counter. However, Bertoni does teach the use of incrementing a counter upon creating a lock upon a resource (col. 5 lines 18-45, "the attribute Reader Count 166 has the value 3 which indicates that three read lock requests have been placed upon this range") and further reducing the count value when unlocking that resource (col. 14 line 61 – col. 15 line 9, "in step 510 the Reader Count for the current sub-lock is decremented by one to indicate that the process requesting the unlock operation no longer desires a read lock on this current sub-lock"). As discussed above, it would have been obvious to include these instructions within the constructor and destructor, as that is where the lock and unlock functions are located. Also, Gallop

teaches conditioning the claiming or relinquishing of critical sections based on the value of a counter and the reference also provides the motivation for doing so (col. 2 lines 33-39, "the monitor transition vector...provides implicit lock counts", "this improves the execution performance by eliminating the need to perform tests to determine if the synchronization comprises a first lock, a shallow lock or a deeply nested lock").

As per claim 18, Gallop discloses a method of managing a resource shared among a plurality of concurrently executing threads, comprising the acts of:

claiming a first critical section, wherein said first critical section is unavailable whenever any of said threads is presently writing to said resource (col. 1 lines 20-49, "because each thread will reserve resources, programming languages such as Java provide mechanisms for serializing critical sections of the program code");

claiming a second critical section, wherein said first critical section is unavailable whenever any of said threads is presently reading from said resource (col. 1 lines 20-49, "because each thread will reserve resources, programming languages such as Java provide mechanisms for serializing critical sections of the program code").

Bertoni further discloses that if a thread is writing to the resource, the thread must wait until the resource is completely unlocked before claiming a lock on that resource (col. 5 lines 46-54, "user desiring a write lock must wait until the range is completely unlocked"). As the critical sections being claimed in this claim are the "write" and

"main" critical sections, the Bertoni reference is applicable since the purpose of what is claimed is to ensure that no other thread is either reading or writing to the resource. Although Bertoni does not specifically teach the use of critical sections of program code, Gallop does and the motivation for combining these references can be found above.

As per claim 21, Bertoni discloses that the access of said resource is a write access (col. 5 lines 13-54, "another user may wish to place an exclusive write lock upon this region"). Bertoni does not specifically disclose that the method further comprises:

relinquishing said second critical section; and

after performing said executing act, relinquishing said first critical section.

However, Gallop does teach the use of serializing critical sections being managed through Java (col. 1 lines 20-60, "Java provide mechanisms for serializing critical sections of the program code"). Since Bertoni does indicate that when a thread is writing to a resource, it should have exclusive access, it would have been obvious to one of ordinary skill in the art to claim both the write and main critical sections for a thread writing to the resource since this would permit it so that no thread can gain a lock on the resource if another is writing to it. This would allow for maintenance of consistent data.

As per claim 22, Bertoni discloses that the access of said resource is a read access (col. 5 lines 13-54, "Mode 154 has the value READ 156 which indicates that the region corresponding to this sub-lock has a read lock placed on it"). Bertoni does not specifically disclose that the method further comprises:

relinquishing said first critical section; and

after performing said executing act, relinquishing said second critical section, unless another set of instructions is presently reading from said resource.

However, Gallop does teach the use of serializing critical sections being managed through Java (col. 1 lines 20-60, "Java provide mechanisms for serializing critical sections of the program code"). Since Bertoni does indicate that when multiple threads are reading from a resource, if one thread should relinquish its lock the others threads may continue to read from the resource, it would have been obvious to one of ordinary skill in the art to relinquish the read critical section only if there are no other threads reading from the resource since this would permit it so that other threads can continue to indicate to the system that the resource is being read from, thus prohibiting another to write to it, possibly compromising data

As per claim 23, Bertoni discloses a method, wherein the determination of whether any set of instructions is presently reading from said resource is made by testing the value of a counter (col. 5 lines 13-54, "a Reader Count of one or above for a read sub-lock indicates that one or more users have desired to place a read lock upon this range").

As per claim 24, neither Gallop nor Bertoni specifically disclose a method, further comprising the acts of:

creating a local class instance; and

after said executing act, destroying said local class instance, and wherein said claiming acts are invoked by the constructor for said local class instance, and wherein the destructor for said local class instance relinquishes at least one of the critical sections.

However, Gallop does discuss the use of Java to implement the method described within the reference. It is well known that Java classes comprise a constructor and destructor. Further, it would have been obvious to claim and relinquish the critical sections within the constructor and destructor, respectively, as discussed above.

As per claim 26, Bertoni discloses a method, further comprising the act of incrementing a counter (col. 6 line 64 – col. 7 line 17, "it is only necessary to increment the Reader Count for Sub-Lock 1"). Bertoni does not specifically teach that the claiming of said second critical section is conditioned upon the value of said counter. However, Gallop does teach the use of critical sections. As the purpose of claiming critical sections is intended to indicate to the system whether or not any thread is using the resource, it would be obvious to one of ordinary skill in the art to condition the claiming of the critical section on the value of the counter. This is because the second critical section is either the write or read critical section. If the thread is attempting to claim the write critical section, and either the write or read counter is non-zero, the thread must wait until both counters have returned to zero in order to ensure that it has exclusive access. Additionally, if attempting to claim the read critical section, the thread must wait if the

write counter is non-zero, or else data may be compromised, and if the read counter is non-zero it is unnecessary to claim the read critical section.

As per claim 27, neither Bertoni nor Gallop specifically discloses claiming and relinquishing a third critical section, wherein said third critical section is relinquished prior to said executing act. However, it would have been obvious to one of ordinary skill in the art to do so, since when performing a write lock it is necessary to ensure that the writing thread has exclusive access to the resource. In order to do so, a writing thread should be able to claim the read critical section since that would ensure that no other thread is reading from the resource. If the read critical section is unavailable, then another thread is reading from the resource, and the writing thread must wait until it has finished in order to ensure that it has exclusive access.

As per claim 30, Gallop discloses a method of managing a resource in a computer environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

in a one of said set of instructions:

opening a local scope (This is inherent in the Java programming language. Code is encapsulated so that classes may be defined in methods, as well as program code being broken up into different functions. Each of these code segments comprises a local scope.);

Gallop does not specifically show creating an object instance within said local scope, wherein said instance comprises or references a constructor method, and wherein said constructor method comprises instructions to obtain a lock on said resource. However, Gallop does show referencing a lock and unlock function in Java (col. 4 line 31 – col. 5 line 22, "the initial monitor transition element 14a is assigned to an object 2...when the object 2 is first created", "the lock transition function 16 is invoked in response to a request to lock an object 2). Further, it would have been obvious to reference the locking function within the constructor of a class, since that would reduce any overhead and simplify code, as well as ensure that the object is properly locked for the length of its execution.

Additionally, Gallop shows:

> performing, subsequent to creating said instance, one or more operations, wherein at least one of said operations reads from or writes to said resource (This is implied by the reference in the sense that the purpose of locking the resource must be to perform some operation upon it); and

Gallop does not specifically show closing said local scope, whereupon said instance is destroyed, said instance further comprising or referencing a destructor method, and wherein said destructor method comprises instructions to release said lock. However, as Gallop's method is to be implemented in Java, it is inherent that upon closing the scope of an object, it would be destroyed and has its destructor called. Further, it would have been obvious to one of ordinary skill in the art to include

instructions to release the lock the object possesses since the object has gone out of scope, there would be no need to occupy the monitor's lock and tie up resources.

As per claims 3, 16, and 31, neither Gallop nor Bertoni specifically disclose that the class instance is a class instance in the C++ programming language. However, the method of Gallop teaches the use of Java, another object-oriented programming languages with many similarities to C++. Class instances in Java also comprise a constructor and destructor, thus the utility found in the applicant's invention could be applied to the Java language as well. Additionally, Java may provide an advantage in the sense that a Thread class is defined, as well a Writer and Reader class, which operate by claiming critical sections of the program code and lock objects in their constructors (see the Java specification on java.sun.com).

As per claims 13 and 32, Gallop does not specifically disclose that said constructor further comprises an instruction to claim a critical section, and wherein said destructor further comprises an instruction to relinquish said critical section. However, Gallop does teach of the utility of claiming critical sections of the program code (col. 1 lines 20-60). Further, it would have been obvious to one of ordinary skill in the art to include a line of code to claim or relinquish a critical section of code along with the lock and unlock functions, respectively as this is the point at which the decision is made whether or not to claim that particular critical section. The motivation for providing

these functions in the constructor and destructor are discussed above, thus it follows that these lines of code would be within the constructor and destructor.

As per claims 7, 8, and 34, Bertoni teaches of a method, wherein constructor further comprises instructions to increment a counter, wherein the value of said counter is the number of read locks outstanding on said resource (col. 5 lines 18-45, "the attribute Reader Count 166 has the value 3 which indicates that three read lock requests have been placed upon this range"). The motivation for including all instructions associated with locking the resource in the constructor is discussed above.

As per claim 37, let it first be noted that the use of critical sections has been well documented and discussed above, and the justifications and motivations for using them within this claim have previously been met.

It follows that Bertoni discloses a method of managing a resource in a computing environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

(a) issuing, in a first set of instructions, a first request for said set of instructions to obtain a lock on said resource, wherein said request comprises an indication as to whether said set of instructions needs a read lock on said resource or a write lock on said resource (col. 5 lines 14-45, "the attribute Mode 154 has the value READ 156 which indicates that the region corresponding to this sub-

lock has a read lock placed upon it. As discussed above, Mode could also be WRITE");

(b) claiming a first critical section (provided by Gallop and discussed above);

(c) if said indication is that said first set of instructions needs a write lock on said resource:

(c)(1) claiming a second critical section (provided by Gallop and discussed above); and

(c)(2) relinquishing said second critical section (provided by Gallop and discussed above);

whereupon said write lock is granted to said first set of instructions (col. 5 lines 13-54, "a sub-lock of type WRITE the Reader Count must always be one which indicates the exclusive nature of a write lock", wherein upon determining that the writing thread would have exclusive access to the resource, it is granted a lock); and

(d) if said indication is that said first set of instructions needs a read lock on said resource:

(d)(1) relinquishing said first critical section (provided by Gallop, wherein there is no need to hold the write critical section if attempting to read from the resource and the ability to claim the write critical section shows that no thread is writing the resource); and

(d)(2) if no other one of said plurality of sets of instructions, exclusive of said first set of instructions, has a read lock on said resource,

claiming said second critical section (provided by Gallop, wherein the thread will claim the read critical section unless another reading thread already has it);

whereupon said read lock is granted to said first set of instructions (col. 4 lines 31-54, col. 5 lines 13-54, "if a user has a write lock applied to a portion of the resource this means that no other user may apply a read or write lock to the same portion of the resource", wherein upon determining that no other thread is writing to the resource, it is permissible to grant a read lock to the requesting thread.

As per claims 6, 14, 19, 20, 33, and 38, Gallop teaches a method, wherein said threads are threads of a single multi-threaded computer program executing under an operating system (col. 4 lines 6-22, "multiple threads running in a single program allow the program to perform multiple tasks at the same time"). Further, Gallop teaches the limitation wherein said critical sections are implemented by way of a critical section facility of said operating system (col. 1 lines 20-49, "because each thread will reserve resources, programming languages such as Java provide mechanisms for serializing critical sections of the program code"). Further, it is well known in the art that Java bytecode must be run on a Java Virtual Machine (JVM), which is essentially a virtual operating system. Since the mechanism for serializing critical sections is described with Java, it can be considered a subset of the Java operating system (JVM) and thus meets the limitation set forth in the claims.

As per claim 39, neither Bertoni nor Gallop specifically discloses:

after said act of issuing said first request, claiming a third critical section; and

before, or contemporaneously with, the granting of a lock, relinquishing said third critical section.

However, it would have been obvious to one of ordinary skill in the art to do so, since when performing a write lock it is necessary to ensure that the writing thread has exclusive access to the resource. In order to do so, a writing thread should be able to claim the read critical section since that would ensure that no other thread is reading from the resource. If the read critical section is unavailable, then another thread is reading from the resource, and the writing thread must wait until it has finished in order to ensure that it has exclusive access.

As per claim 40, Bertoni shows a method, further comprising the acts of:

(e) issuing, in said first set of instructions, a second request to release said lock (col. 15 lines 23-43, "once process S has finished with its region, it may release its read lock upon the region);

(f) if said lock is a read lock and no other one of said sets of instructions, exclusive of said first set of instructions, presently has a read lock on said resource, relinquishing said second critical section (provided by Gallop, wherein if another thread is reading from the resource, the ownership of the read critical

section must be passed along, so the system knows that the resource is occupied); and

(g) if said lock is a write lock, relinquishing said first critical section (provided by Gallop, wherein upon completion of writing to the resource, the write lock must be freed so that other threads know the resource is available).

As per claim 41, Gallop discloses a method further comprising the acts of:

creating a local class instance; and

wherein said first request is issued by the constructor for said class instance (col. 4 lines 31-49, "when the object 2 is initially created the monitor transition element field 3 in the object 2 stores the address or pointer 12 to the first monitor transition element 14a...each monitor transition element 14 is implemented as a structure containing two function pointers: &lock 16 and &unlock 18"). Further, since the mechanism is designed to be implemented in Java, when a class instance is created, a constructor for that class is called, and this is well known in the art. It would be obvious to one of ordinary skill in the art to combine the teachings of Gallop with those of Bertoni since the combination of these references provides a sound means of locking a resource. By using the framework of Gallop, a data structure is defined that allows encapsulation of threads and incorporates the locking and unlocking functions into that data structure. To limit this framework, the method of Bertoni ensures that no thread is reading or writing to the resource while another thread is writing to the resource. In doing so, data can be assured

to be accurate, whereas if a thread was reading from the resource at the same time another was writing, the data may be compromised and could not be considered accurate.

Gallop does not specifically disclose:

destroying said local class instance;

wherein said second request is issued by the destructor for said class instance. However, Gallop does teach that the release of a lock be performed through the use of an unlock function (col. 4 lines 59-62, "the unlock transition function 18 is invoked in response to a request to unlock an object"). It would be obvious to reference this function within the class's destructor since at the point when an object goes out of scope, it would be senseless to have a thread continue to possess an object's lock when it no longer exists. Thus, in addition to referencing the "&unlock" function in Gallop whenever a thread desires to release a lock, it would be obvious to one of ordinary skill in the art to also reference the "&unlock" function when the object goes out of scope and the destructor is called.

As per claims 25 and 42, neither Gallop nor Bertoni specifically disclose a method, wherein said local class instance is a C++ class, wherein said act of creating a local class instance comprises opening a local scope in a program in the C++ programming language, and wherein said act of destroying said local class instance comprises closing said local scope.

However, the method of Gallop teaches the use of Java, another object-oriented programming languages with many similarities to C++. Classes in Java are also created

and destroyed by opening and closing local scopes, respectively (see the Java specification on java.sun.com). Thus, the utility found therein is also met by the Java programming language, as disclosed in Gallop.

As per claim 43, Bertoni discloses a method further comprising the acts of incrementing and decrementing a counter, wherein the value of said counter is the number of read locks outstanding on said resource, and wherein said determinations of whether any other of said sets of instructions has a read lock on said resource are made by testing the value of said counter (col. 5 lines 13-54, "a Reader Count of one or above for a read sub-lock indicates that one or more users have desired to place a read lock upon this range", wherein for each read lock applied to a region, the counter is incremented and when that thread is finished with the resource the counter is decremented).

As per claims 17, 28, 35, and 44, Bertoni discloses that said resource comprises a data object located within the address space of said computer program (col. 4 lines 61-67, "The resource 20 could be any resource within a computer system. By way of example, the resource 20 could be a file, a portion of a file, an object...or any other resource or media used by a computer system or network that is accessible by a user").

As per claims 29, 36, and 45, a computer-readable medium having computer-executable instructions to perform the method is inherent in the disclosure of Bertoni as

the method described is intended for implementation on a computer system. Thus, the system must have a medium with computer-executable instructions.

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Thomas et al.	USPN 5,931,919
Joy	USPN 5,761,670
Haber et al.	USPN 4,435,766
Holdsworth et al.	USPN 6,052,731
Shi et al.	USPN 5,623,659
Schmuck et al.	USPN 6,032,216
Anfindsen	USPN 5,983,225
McClaughry et al.	USPN 5,933,825
Johnson	USPN 5,721,943
Vartti et al.	USPN 5,678,026

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Syed J Ali whose telephone number is (703) 305-8106. The examiner can normally be reached on Mon-Fri 8-5:30, 1st Friday off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John A Follansbee can be reached on (703) 305-8498. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

November 6, 2002

MAJID A. BANANKHAH
PRIMARY EXAMINER